Test case Design Techniques

Types of Test case Design Techniques:

1) Error Guessing

2) Equivalence Class Partitioning

3) Boundary Value Analysis

4) Decision Table Technique

5) State Transition Diagram

1) **Error Guessing** – Here, TE will guess the error and derive more scenarios.

Example: Assume there is an “Amount” text field and the requirement says that it accepts + integer only.

Now we will have to enter only invalid inputs such as -10, abc, 10@ etc and try to guess more errors in this case.

2) **Equivalence Class Partitioning** - Here, when the input is in range of values, let us say, there is an “Amount” text field and the requirement says that it can accept numbers between 100 to 500. In this case what we can do is instead of entering all numbers, we can divide this range into equal classes I,e,. -100 to 0, 0 to 100, 100 to 200, 200 to 300, 300 to 400, 400 to 500 & 500 to 600. After this try entering any 1 value from each of this class. Example if you enter 150 for the class 100 to 200, need not enter other values in range of 100 to 200, if it is accepting, then test case is pass, if it is not accepting, then test case is fail. Likewise we can only test only 2 scenarios/values(which includes 5 positive & 2 negative scenarios) and ensure that our test case coverage is achieved.

3) **Boundary Value Analysis** – Lets say, here you need values between the range A to B. We need tests for A,A+ & A- similarly B, B+ & B- . So here we will have 4 positive scenarios and 2 negative scenarios. Since the boundaries are covered, our test case coverage is good and no need to test for other values.

4) **Decision Table Technique** – In this Technique, we check for multiple conditions, combinations and Rule criterias.

Formula = 2^no of conditions= total no of rules or scenarios

Example1:

Requirement- Customer wants to order from Swiggy,

1) First time customers get 50% discount

2) If the coupon code is used, they get a 25% discount.

In this case there are 2 conditions, 2^2 = 4 scenarios we can derive.

Example 2 for Login page:

5) **State Transition Diagram** – This technique is used to check for different screens of a software. It is basically a pictorial representation of the scenarios.

Example: Let's say a person has to withdraw cash from an ATM machine, we can derive 4 scenarios for this.

1) When he enters the correct pin for the first time, he withdraws the cash.

2) When he enters the wrong pin for the first time and the correct pin for the second time, he can withdraw the cash.

3) When he enters the wrong pin for the first and second time and enters the correct pin for the third time, he can withdraw the cash.

4) When he enters the wrong pin for all attempts I,e,. first, second & third attempt, the card gets blocked and he cannot withdraw the cash.

These scenarios customers can write in a pictorial way and present how scenarios we can derive here.

Orthogonal Array Testing is a systematic, statistical method used in software testing to reduce the number of test cases needed while still effectively covering various scenarios. It's particularly useful in situations where exhaustive testing is impractical due to time or resource constraints. Here are some key notes on orthogonal array matrix in test case design:

1. Purpose: The primary goal of using orthogonal array testing is to achieve

maximum test coverage with a minimal number of test cases.

2. Reduction in Test Cases: By carefully selecting an appropriate orthogonal array, the number of test cases required can be significantly reduced compared to exhaustive testing, saving time and resources.
3. Efficiency: Orthogonal array testing is efficient because it allows testers to focus on critical scenarios and combinations while avoiding redundant or irrelevant tests.
4. Designing Orthogonal Arrays: Designing an orthogonal array involves selecting the appropriate array size (number of columns), determining the factors (parameters) to be tested, and assigning levels to each factor. Tools and mathematical techniques are often employed to generate orthogonal arrays.
5. Validation and Verification: After executing the test cases derived from orthogonal arrays, it's crucial to validate the results and verify that the system behaves as expected under various conditions.

In summary, orthogonal array matrix in test case design is a powerful technique for optimizing test coverage while minimizing the number of test cases required, making it a valuable tool in software testing methodology.

Traceability matrix

It is also called as Requirement Traceability matrix(RTM) or Cross reference matrix (CRM)

It is a document through which we are ensuring that each and every requirement has minimum test cases.

Advantages of RTM:

1) Ensures complete requirements are documented.

2) It helps in analyzing the root cause of any defect.

3) Applications can be developed according to the requirements.

Types of RTM:

1) Forward Traceability Matrix

2) Backward Traceability Matrix
3) Bi-directional Traceability Matrix

1) **Forward Traceability Matrix** – It is used to map requirements to the test cases. This is done before Test case execution. Here we are ensuring that the product developments are going in the right direction.

2) **Backward Traceability Matrix** – It is used to map Test cases to the requirement. This is done after Test case execution. Here we are ensuring that we are not going against/developing products against customer requirements.

3) **Bi-directional Traceability Matrix** – It is a combination of both Forward Traceability Matrix and Backward Traceability Matrix.